PICO
THE RISE OF MICROCONTROLLER
RUBY

EPISODE Ⅳ

@hasumikin

RubyConf Taiwan 2023

Dec. 15, 2023

Arts-Wallpapers.com

Find PicoRuby's history on the internet

EPISODE Ⅱ

ATTACK OF THE RAKE

EPISODE Ⅲ

REVENGE OF THE STDIN

Extra edition in Euruko 2023

"A Beginner's Complete Guide"

Arts-Wallpapers.com

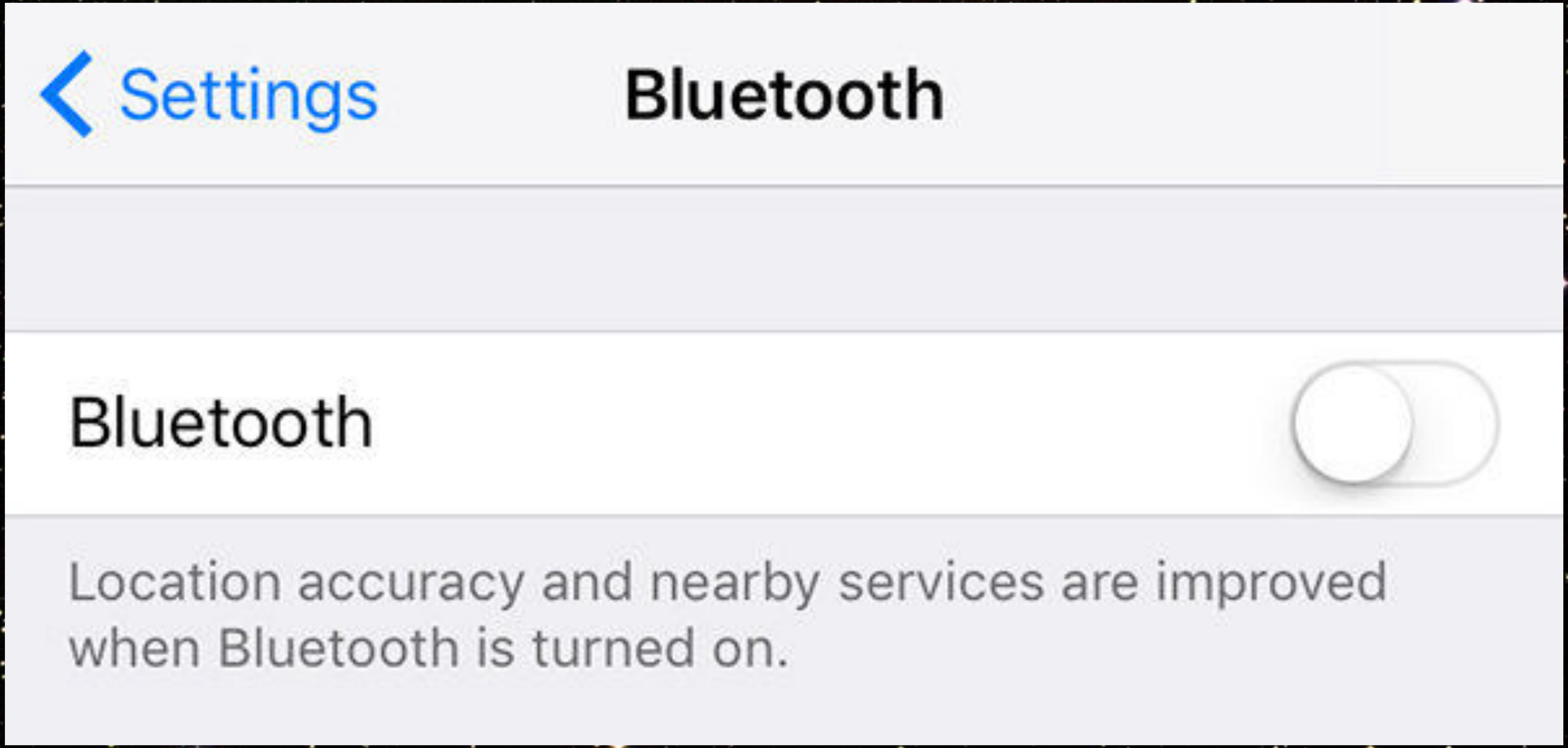
self.inspect

- Hitoshi HASUMI
- hasumikin (GitHub ,ex-Twitter, Bluesky and Mastodon)
- Creator of PicoRuby and PRK Firmware
- Committer of CRuby's IRB and Reline
- First prize of Fukuoka Ruby Award (2020 and 2022✌️)
- A final nominee of Ruby Prize 2021

Arts-Wallpapers.com

I have a favor to ask

If possible, please turn off your Bluetooth



Bluetooth

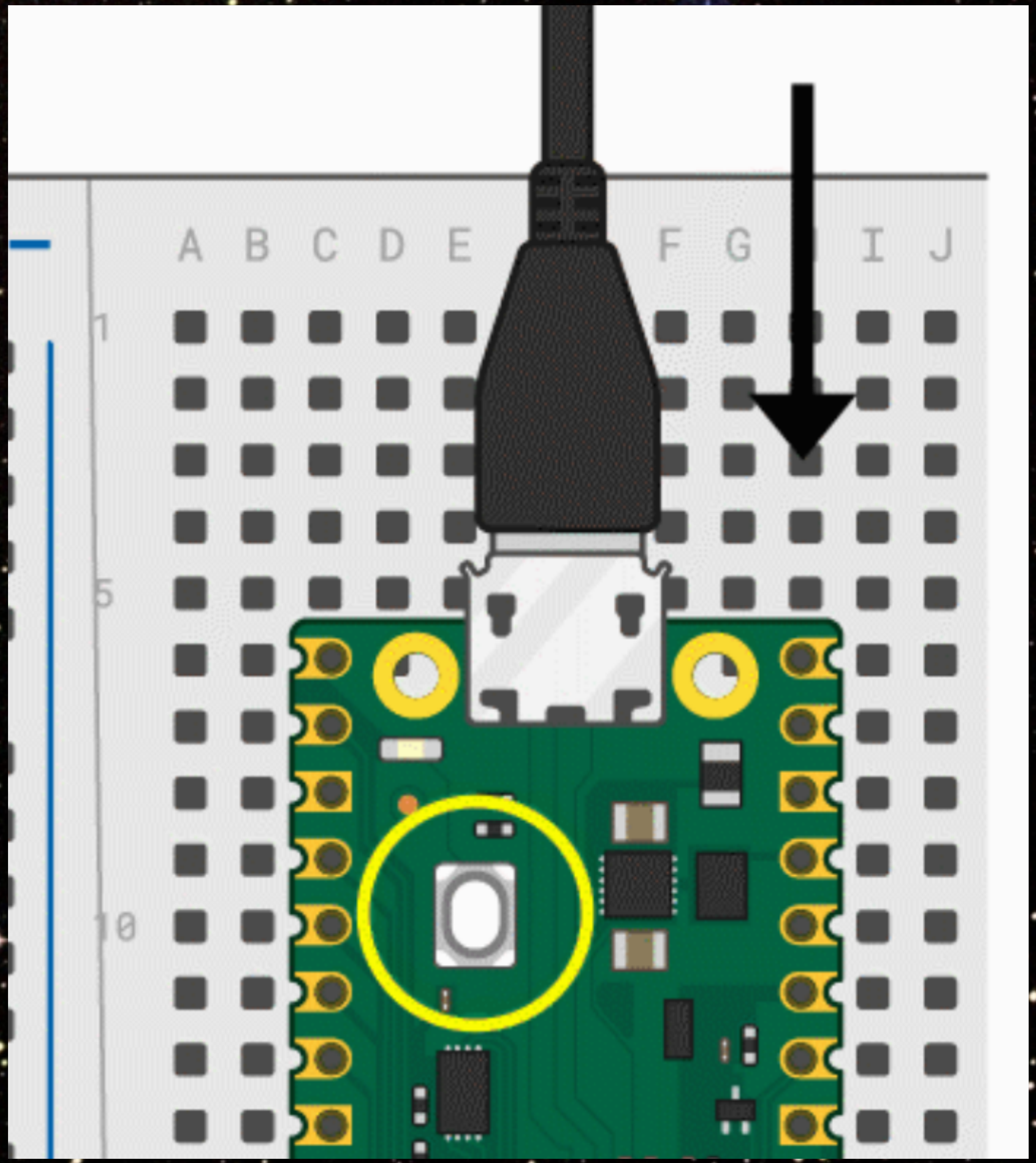
Location accuracy and nearby services are improved when Bluetooth is turned on.

To avoid channel congestion during my talk

Arts-Wallpapers.com

Let's begin 1/5

- Connect Pi Pico and PC while pressing the BOOTSEL button
  - You'll find "RPI-RP2" drive in file manager



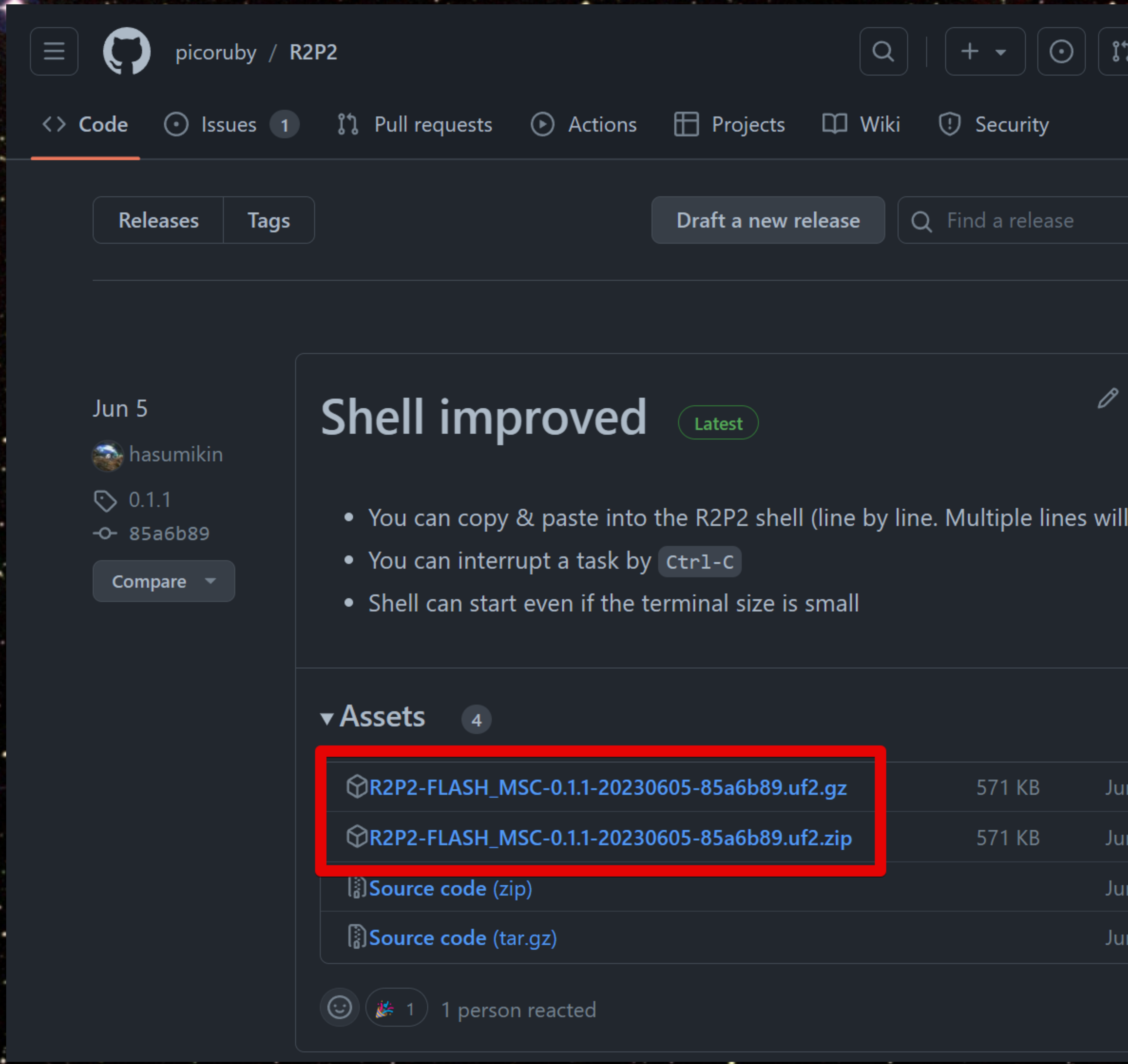
https://www.raspberrypi.org/documentation/rp2040/getting-started

Arts-Wallpapers.com

Let's begin 2/5

- Download the latest `R2P2-*.uf2` from GitHub

https://github.com/picoruby/R2P2/releases



Jun 5
hasumikin
0.1.1
85a6b89

Shell improved Latest

- You can copy & paste into the R2P2 shell (line by line. Multiple lines will
- You can interrupt a task by `Ctrl-C`
- Shell can start even if the terminal size is small

Assets 4

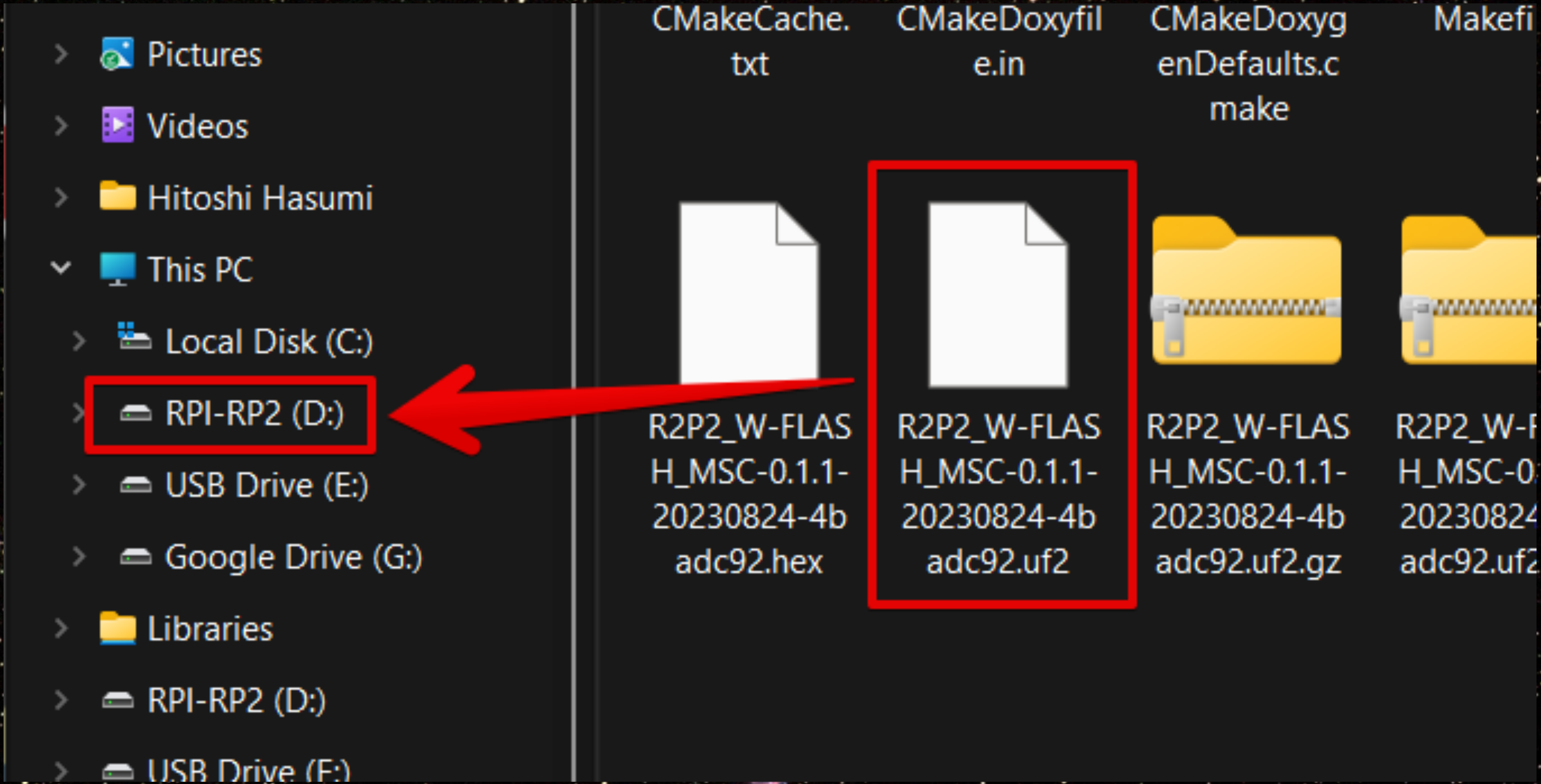
R2P2-FLASH_MSC-0.1.1-20230605-85a6b89.uf2.gz	571 KB	Ju
R2P2-FLASH_MSC-0.1.1-20230605-85a6b89.uf2.zip	571 KB	Ju
Source code (zip)		Ju
Source code (tar.gz)		Ju

1 1 person reacted

Arts-Wallpapers.com

Let's begin 3/5

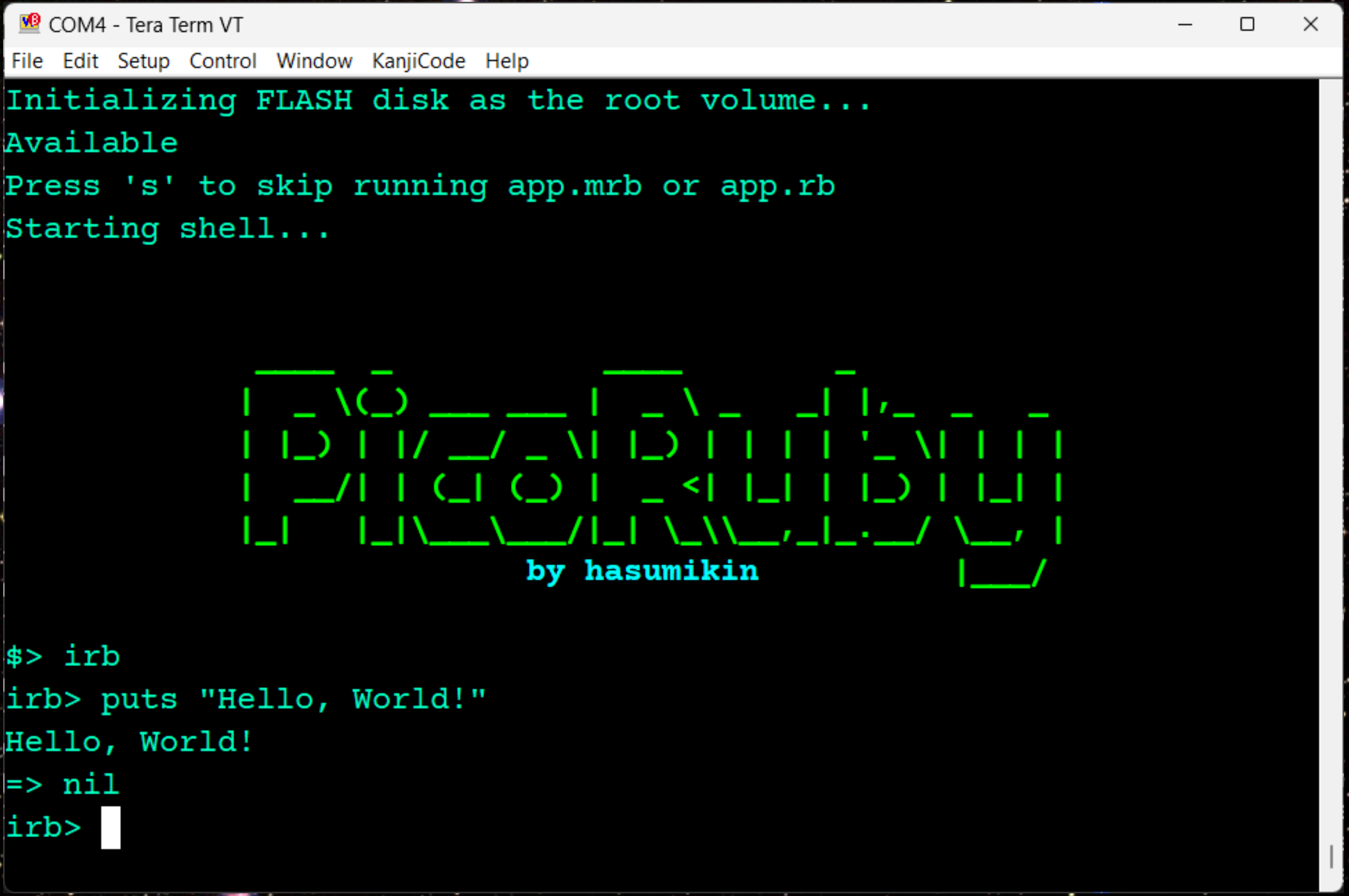
- Drag & drop `R2P2-*.uf2` into RPI-RP2 drive


Pictures
Videos
Hitoshi Hasumi
This PC
Local Disk (C:)
RPI-RP2 (D:)
USB Drive (E:)
Google Drive (G:)
Libraries
RPI-RP2 (D:)
USB Drive (E:)
CMakeCache.txt
CMakeDoxyfile.in
CMakeDoxygenDefaults.cmake
Makefi
R2P2_W-FLASH_MSC-0.1.1-20230824-4badc92.hex
R2P2_W-FLASH_MSC-0.1.1-20230824-4badc92.uf2
R2P2_W-FLASH_MSC-0.1.1-20230824-4badc92.uf2.gz
R2P2_W-FLASH_MSC-0.1.1-20230824-4badc92.uf2

Arts-Wallpapers.com

Let's begin 4/5

- Open a proper serial port on terminal emulator

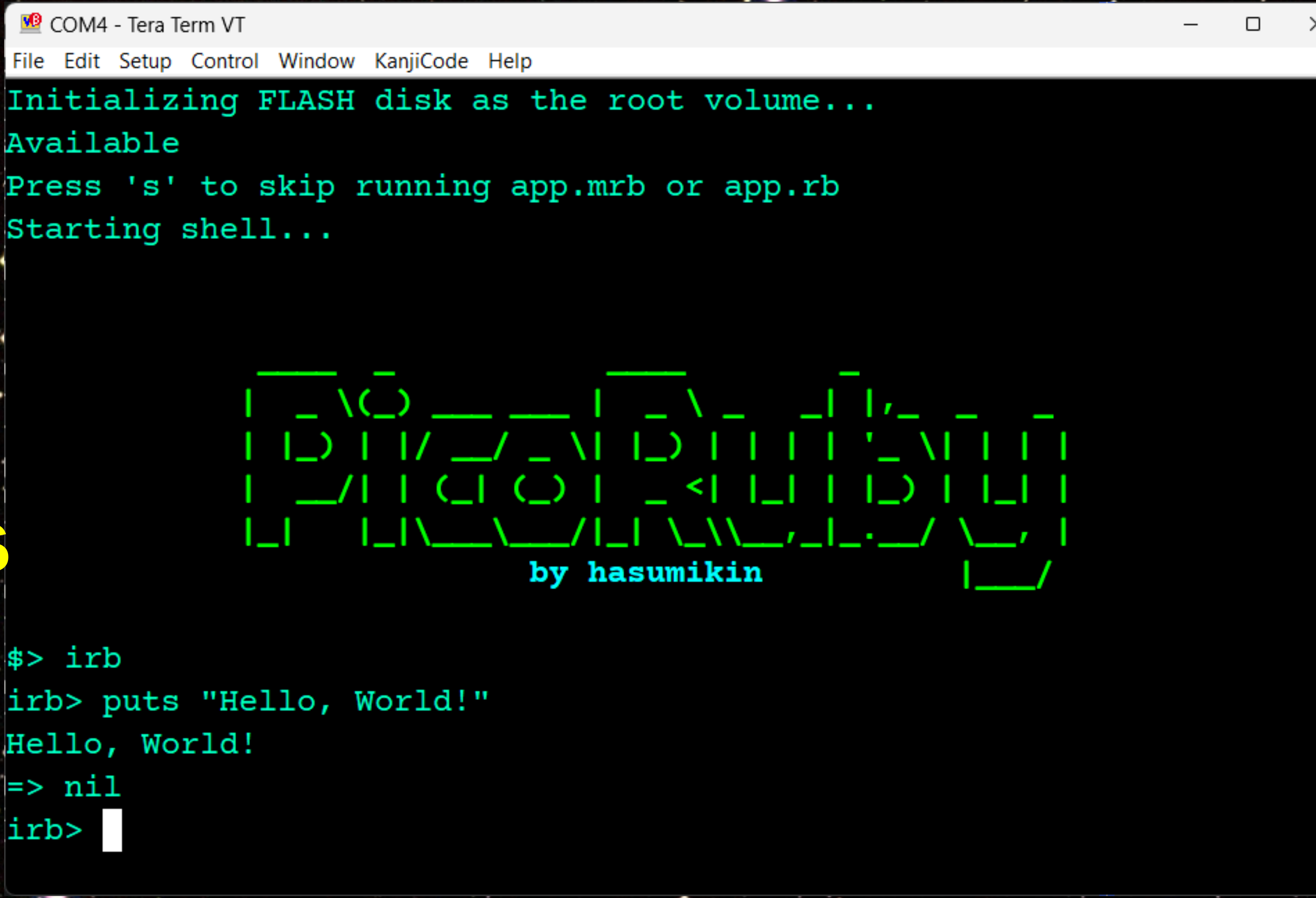


```
Initializing FLASH disk as the root volume...
Available
Press 's' to skip running app.mrb or app.rb
Starting shell...


             ____  _           ____        _
            |  _ \(_) ___ ___ |  _ \ _   _| |__  _   _
            | |_) | |/ __/ _ \| |_) | | | | '_ \| | | |
            |  __/| | (_| (_) |  _ <| |_| | |_) | |_| |
            |_|   |_|\___\___/|_| \_\\__,_|_.__/ \__, |
                           by hasumikin          |___/

$> irb
irb> puts "Hello, World!"
Hello, World!
=> nil
irb>
```

Arts-Wallpapers.com

Let's begin 5/5

- R2P2 is a Unix-like shell running on Pi Pico
- You can use some commands like `cd`, `ls`, `mkdir`, and `irb`
- Every computing process is happening on Pi Pico



```
Initializing FLASH disk as the root volume...
Available
Press 's' to skip running app.mrb or app.rb
Starting shell...


     ____  _           ____        _
    |  _ \(_) ___ ___ |  _ \ _   _| |__  _   _
    | |_) | |/ __/ _ \| |_) | | | | '_ \| | | |
    |  __/| | (_| (_) |  _ <| |_| | |_) | |_| |
    |_|   |_|\___\___/|_| \_\\__,_|_.__/ \__, |
                   by hasumikin          |___/

$> irb
irb> puts "Hello, World!"
Hello, World!
=> nil
irb>
```

Arts-Wallpapers.com

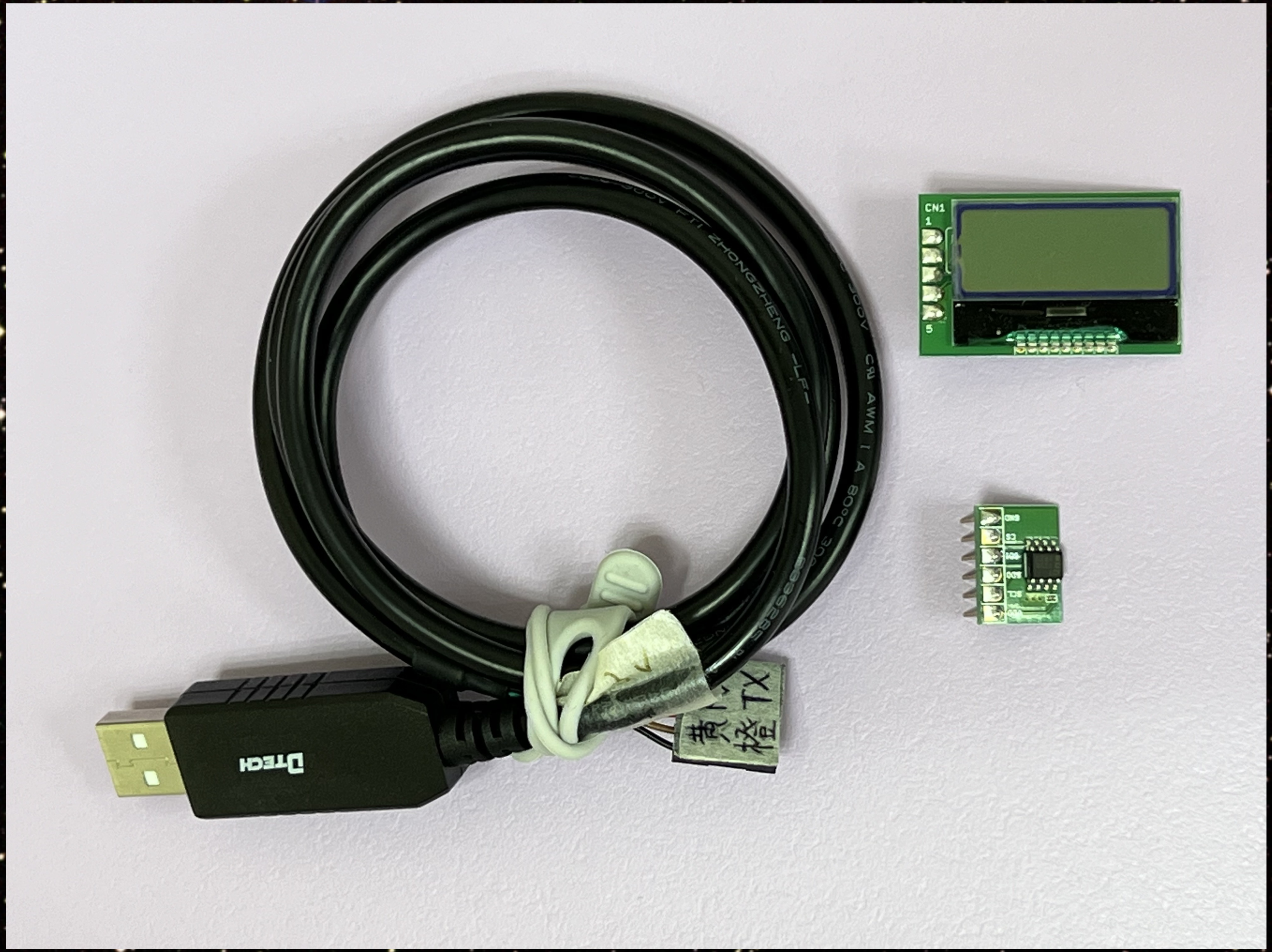
By the way, R2P2 stands for

Ruby on
Raspberry
Pi
Pico

Arts-Wallpapers.com

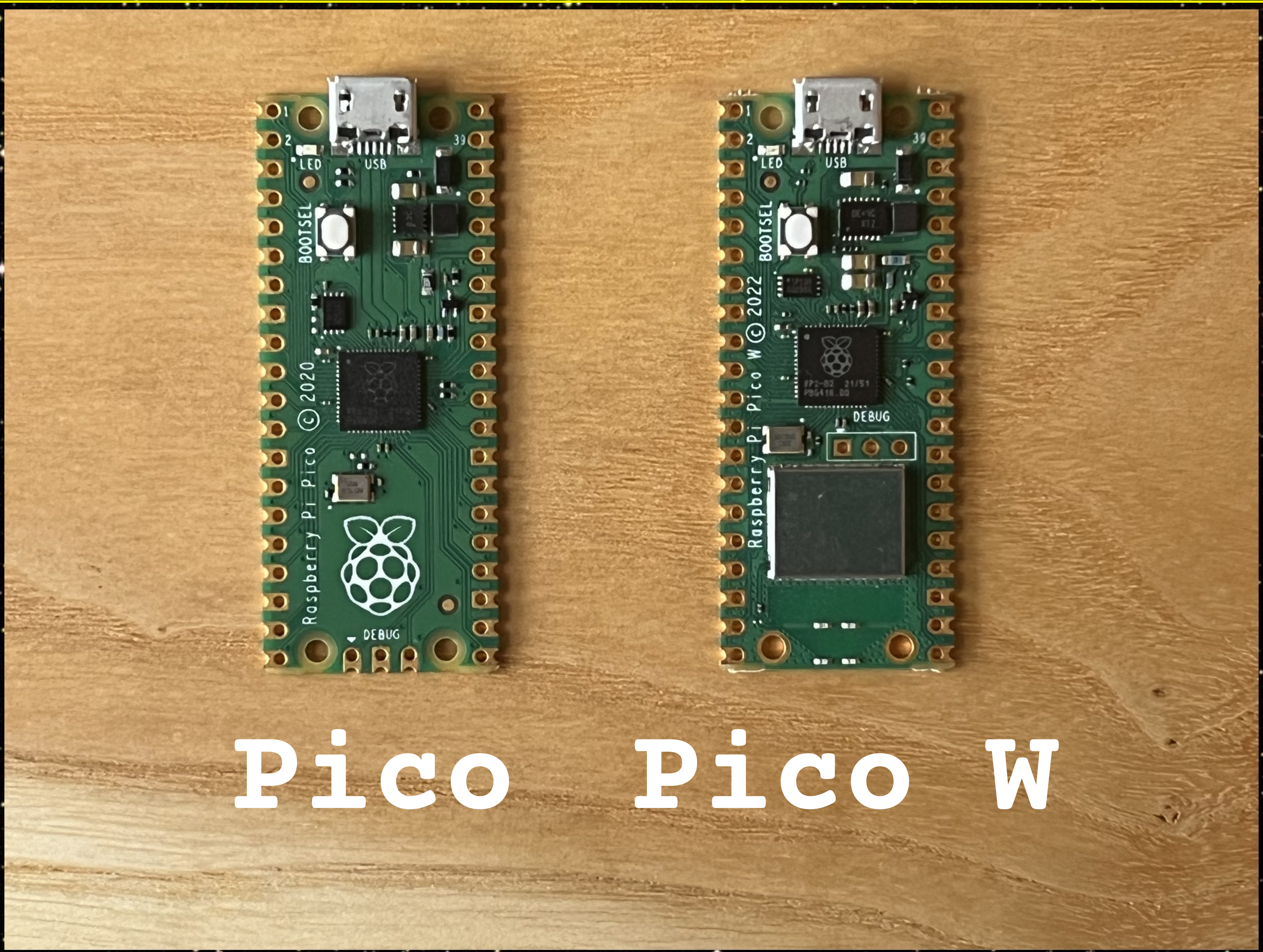
Today's setup

- Raspberry Pi Pico W * 2
- AQM0802 charactor display module
- ADT7310 thermo sensor module
- FTDI USB-TTL serial coverter cable



Arts-Wallpapers.com

Raspberry Pi "Pico" and "Pico W"



Arts-Wallpapers.com

GPIO

```
irb> gpio = CYW43::GPIO.new(CYW43::GPIO::LEĐ_PIN)
   # gpio = GPIO.new(25, GPIO::OUT) # If Pico w/o W
irb> gpio.write 1   #=> LEĐ turns on
irb> gpio.write 0   #=> LEĐ turns off
irb> 3.times do
irb*   gpio.write 1
irb*   sleep 1
irb*   gpio.write 0
irb*   sleep 1
irb* end            #=> LEĐ blinks three times
```

Arts-Wallpapers.com

LED wraps GPIO

```ruby
# /lib/led.rb in R2P2 drive
require 'gpio'
require 'cyw43'
CYW43.init
class LED
  def initialize
    @gpio = CYW43::GPIO.new(CYW43::GPIO::LED_PIN)
    #     = GPIO.new(25, GPIO::OUT) # If Pico w/o W
  end
  def on
    @gpio.write 1
  end
  def off
    @gpio.write 0
  end
  def invert
    @gpio.read == 1 ? off : on
  end
end
```

Arts-Wallpapers.com

LED wraps GPIO

```
irb> require ’led’
irb> led = LEĐ.new
irb> 3.times do
irb*   led.invert
irb*   sleep 1
irb* end
```

Arts-Wallpapers.com

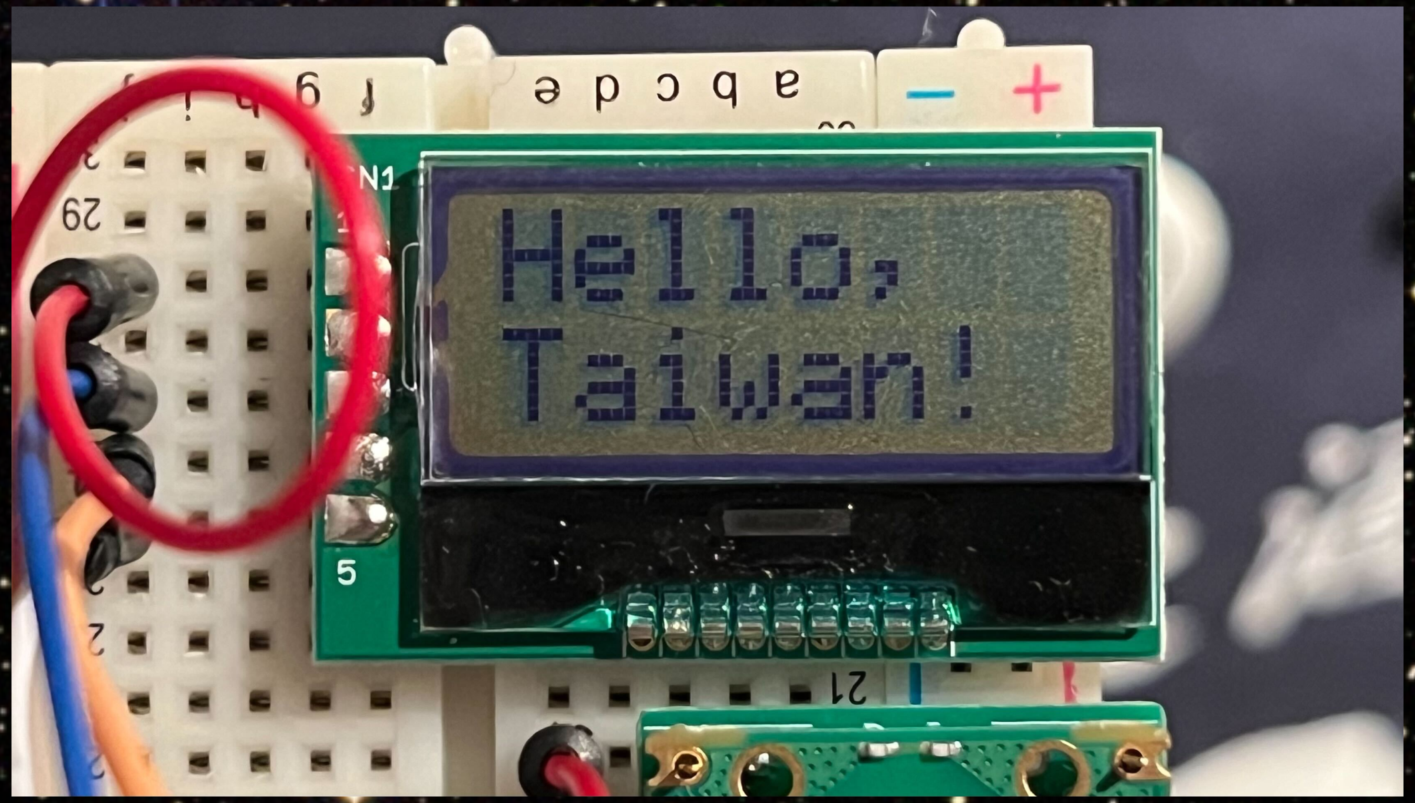
Peripherals for serial communication

- I2C: To communicate between integrated circuits with support for multiple devices connected to the same bus
- SPI: To facilitate high-speed communication between microcontrollers and peripheral devices
- UART: To establish asynchronous serial communication between devices

Arts-Wallpapers.com

I2C

```
irb> require 'i2c'
irb> i2c = I2C.new(unit: :RP2040_I2C1, sda_pin: 26, scl_pin: 27)
irb> [0x38, 0x39, 0x14, 0x70, 0x56, 0x6c].each { |i| i2c.write(0x3e, 0, i); sleep_ms 1 }
irb> [0x38, 0x0c, 0x01].each { |i| i2c.write(0x3e, 0, i); sleep_ms 1 }
irb> "Hello,".bytes.each { |c| i2c.write(0x3e, 0x40, c); sleep_ms 1 }
irb> i2c.write(0x3e, 0, 0x80|0x40)
irb> "Taiwan!".bytes.each { |c| i2c.write(0x3e, 0x40, c); sleep_ms 1 }
```



Arts-Wallpapers.com

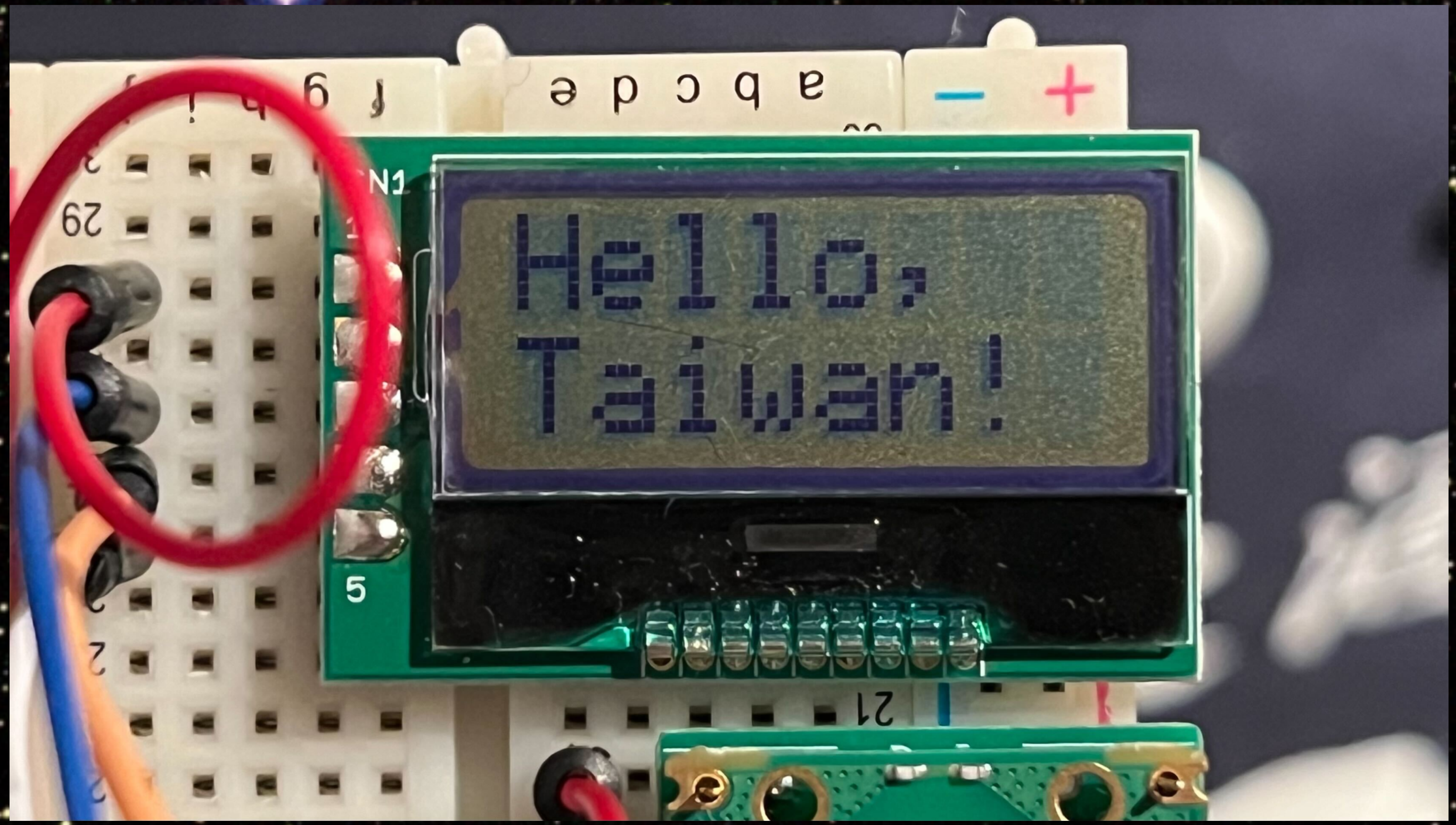
LCD wraps I2C

```ruby
# /lib/lcd.rb in R2P2 drive
require 'i2c'
class LCD
  ADDRESS = 0x3e # 0x7c == (0x3e << 1) + 0 (R/W)
  def initialize(i2c:)
    @i2c = i2c
    reset
  end
  def reset
    [0x38, 0x39, 0x14, 0x70, 0x56, 0x6c].each { |i| @i2c.write(ADDRESS, 0, i) }
    sleep_ms 200
    [0x38, 0x0c, 0x01].each { |i| @i2c.write(ADDRESS, 0, i) }
  end
  def putc(c)
    @i2c.write(ADDRESS, 0x40, c)
    sleep_ms 1
  end
  def print(line)
    line.bytes.each { |c| putc c }
  end
# ...
# See https://github.com/picoruby/picoruby/tree/master/mrbgems/picoruby-ble/example/broadcaster-observer
```

Arts-Wallpapers.com

LCD wraps I2C

```
irb> require 'lcd'
irb> lcd = LCD.new(i2c: I2C.new(unit: :RP2040_I2C1, sda_pin: 26, scl_pin: 27))
irb> lcd.print "Hello,"
irb> lcd.break_line
irb> lcd.print "Taiwan!"
```



Arts-Wallpapers.com

SPI

```
irb> require 'spi'
irb> spi = SPI.new(unit: :RP2040_SPI0, cipo_pin: 16,
           cs_pin: 17, sck_pin: 18, copi_pin: 19)
irb> spi.select
irb> spi.write(255,255,255,255) # Reset
irb> spi.write(0x54)            # Start continuous mode
irb> data = spi.read(2).bytes
irb> temp = data[0] << 8 | data[1]
irb> temp / 128.0               # Convert to Celsius
=> 19.5621
```

Arts-Wallpapers.com

THERMO wraps SPI

```ruby
# /lib/thermo.rb in R2P2 drive
require 'spi'
class THERMO
  def initialize(unit:, sck_pin:, cipo_pin:, copi_pin:, cs_pin:)
    @spi = SPI.new(unit: unit, frequency: 500_000, mode: 0, cs_pin: cs_pin,
      sck_pin: sck_pin, cipo_pin: cipo_pin, copi_pin: copi_pin
    )
    @spi.select
    @spi.write 0xFF, 0xFF, 0xFF, 0xFF # Reset
    @spi.write 0x54 # Start continuous mode
    sleep_ms 240
  end

  def read
    data = @spi.read(2).bytes
    temp = (data[0] << 8 | data[1]) >> 3
    # If it minus?
    temp -= 0x2000 if 0 < temp & 0b1_0000_0000_0000
    temp / 16.0  # Convert to Celsius
  end
end
# See https://github.com/picoruby/picoruby/tree/master/mrbgems/picoruby-ble/example/broadcaster-observer
```

Arts-Wallpapers.com

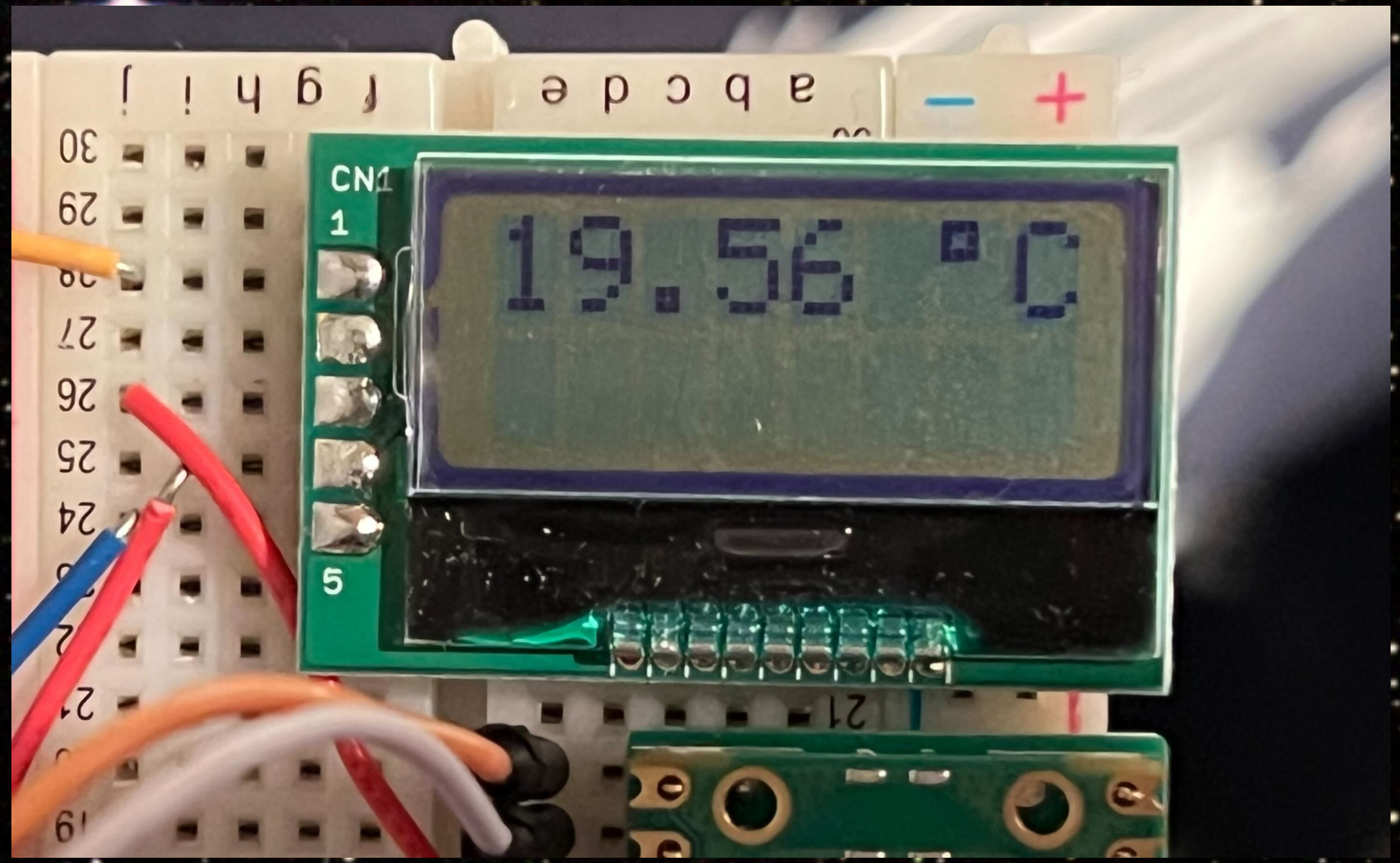
THERMO wraps SPI

```
irb> require 'thermo'
irb> thermo = THERMO.new(unit: :RP2040_SPI0,
              cipo_pin: 16, cs_pin: 17, sck_pin: 18, copi_pin: 19)
irb> thermo.read
=> 19.5621
```

Arts-Wallpapers.com

LCD and THERMO

```
irb> require 'lcd'
irb> lcd = LCD.new(i2c: I2C.new(unit: :RP2040_I2C1, sda_pin: 26, scl_pin: 27))
irb> require 'thermo'
irb> thermo = THERMO.new(unit: :RP2040_SPI0,
                cipo_pin: 16, cs_pin: 17, sck_pin: 18, copi_pin: 19)
irb> lcd.print sprintf("%5.2f \xdfC", thermo.read)
```



Arts-Wallpapers.com

UART

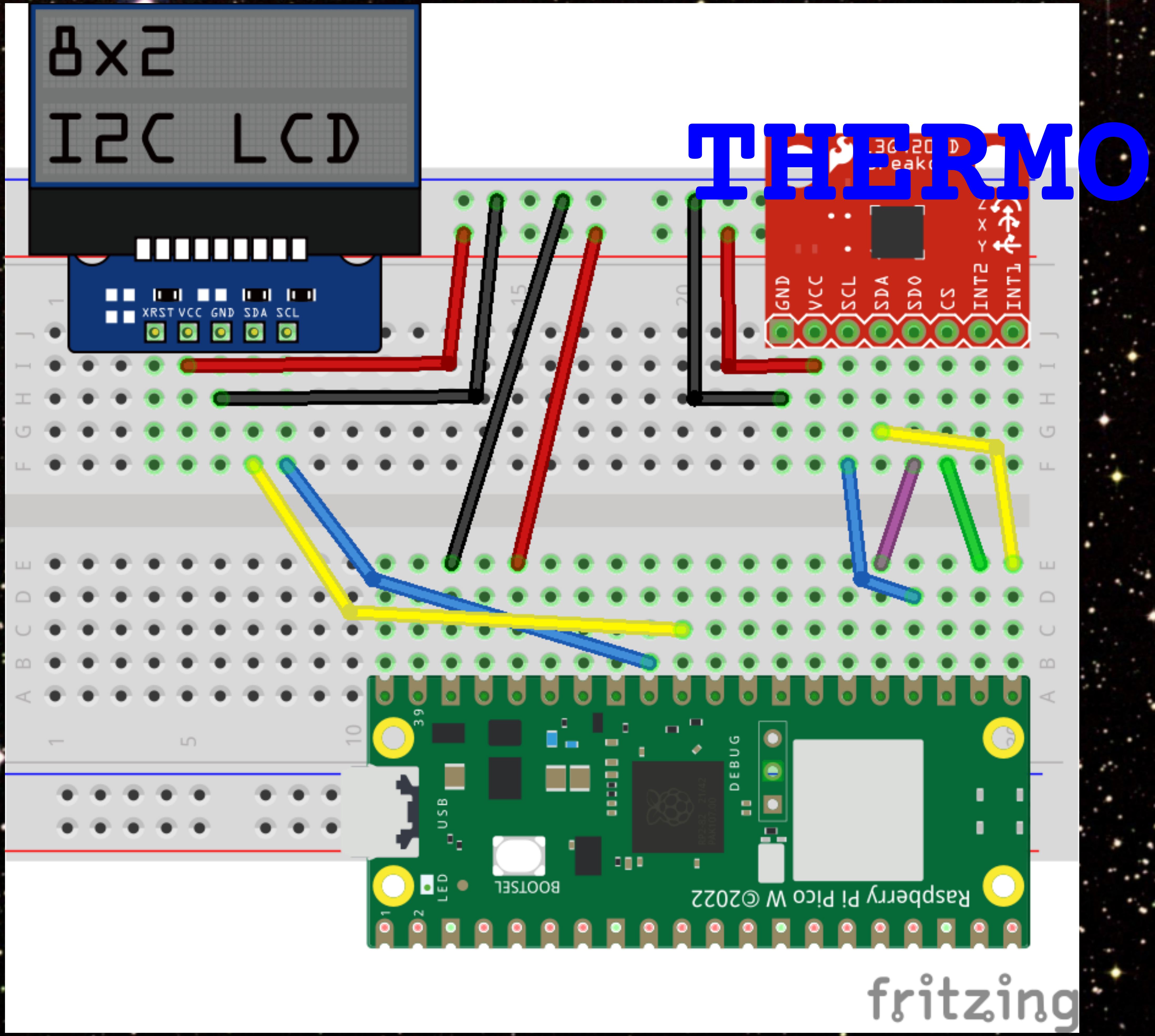
```
irb> require 'uart'
irb> uart = UART.new(unit: :RP2040_UART0, txd_pin: 16, rxd_pin: 17, baudrate: 115200)
irb> uart.puts "Hello from Pico!"
=> nil
irb> while true
irb*   if c = uart.read
irb*     uart.write c    # Echo back
irb*     print c
irb*   end
irb* end
Hello to Pico!           # <= "Hello to Pico!" in UART device
```

You don't need to wrap UART. Use as it is

Arts-Wallpapers.com

BLE Broadcaster

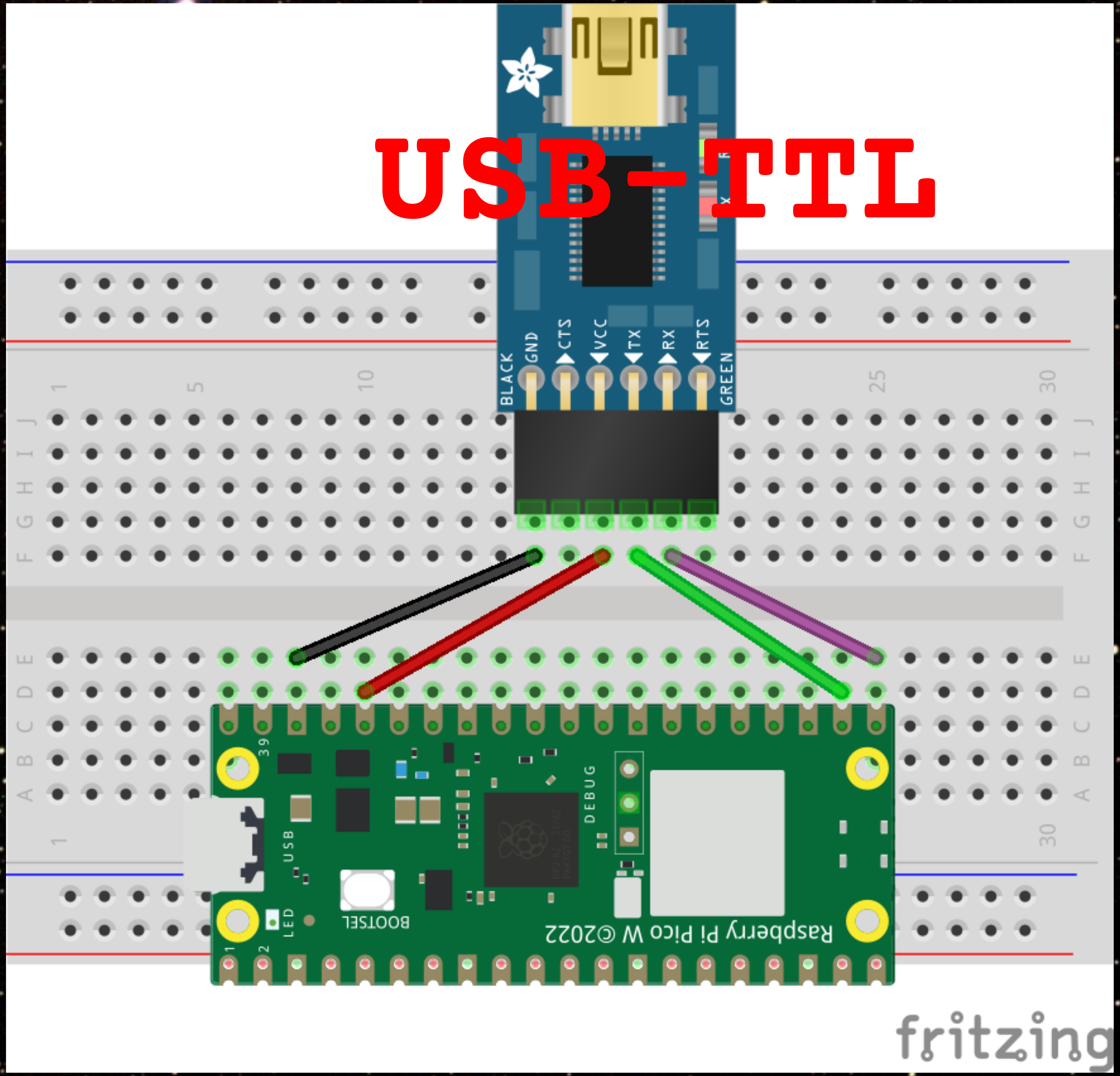
- Pin assign
  - 3V3 to {LCD,THERMO}:VCC
  - GND to {LCD,THERMO}:GND
  - GPIO16 to THERMO:SDO
  - GPIO17 to THERMO:CS
  - GPIO18 to THERMO:SCL
  - GPIO19 to THERMO:SDI
  - GPIO26 to LCD:SDA
  - GPIO27 to LCD:SCL



Arts-Wallpapers.com

BLE Observer

- Pin assign
  - 3V3 to USB-TTL:VCC
  - GND to USB-TTL:GND
  - GPIO16 to USB-TTL:RX
  - GPIO17 to USB-TTL:TX



Arts-Wallpapers.com

Gems in Broadcaster and Observer

```
# in Broadcaster
$> ls /lib
lcd.rb
thermo.rb

# in Observer
$> ls /lib
led.rb
```

Arts-Wallpapers.com

BLE Broadcaster

```ruby
# /home/app.rb in R2P2 drive
class DemoBroadcaster < BLE::Broadcaster
  def initialize
    super(nil)
    led = CYW43::GPIO.new(CYW43::GPIO::LED_PIN)
    @lcd = LCD.new(i2c: I2C.new(unit: :RP2040_I2C1, sda_pin: 26, scl_pin: 27))
    @thermo = THERMO.new(unit: :RP2040_SPI0, cipo_pin: 16, cs_pin: 17, sck_pin: 18, copi_pin: 19)
    @counter = 0
  end
  def adv_data(send_data)
    BLE::AdvertisingData.build do |adv|
      adv.add(0x01, 0xFF)
      adv.add(0x09, "PicoRuby")
      adv.add(0xFF, send_data)
    end
  end
  def heartbeat_callback
    Machine.using_delay do
      temperature = @thermo.read
      @lcd.clear
      @lcd.print sprintf("%5.2f \xdfC", temperature)
      @lcd.break_line
      @lcd.print "+" * (@counter % 9)
      advertise(adv_data (temperature * 100).to_i.to_s)
    end
    @counter += 1
    Watchdog.update
  end
  def packet_callback(event_packet)
    case event_packet[0]&.ord # event type
    when 0x60 # BTSTACK_EVENT_STATE
      return unless event_packet[2]&.ord ==  BLE::HCI_STATE_WORKING
      @state = :HCI_STATE_WORKING
      Watchdog.enable(2000)
    end
  end
end
DemoBroadcaster.new.start
# See https://github.com/picoruby/picoruby/tree/master/mrbgems/picoruby-ble/example/broadcaster-observer
```

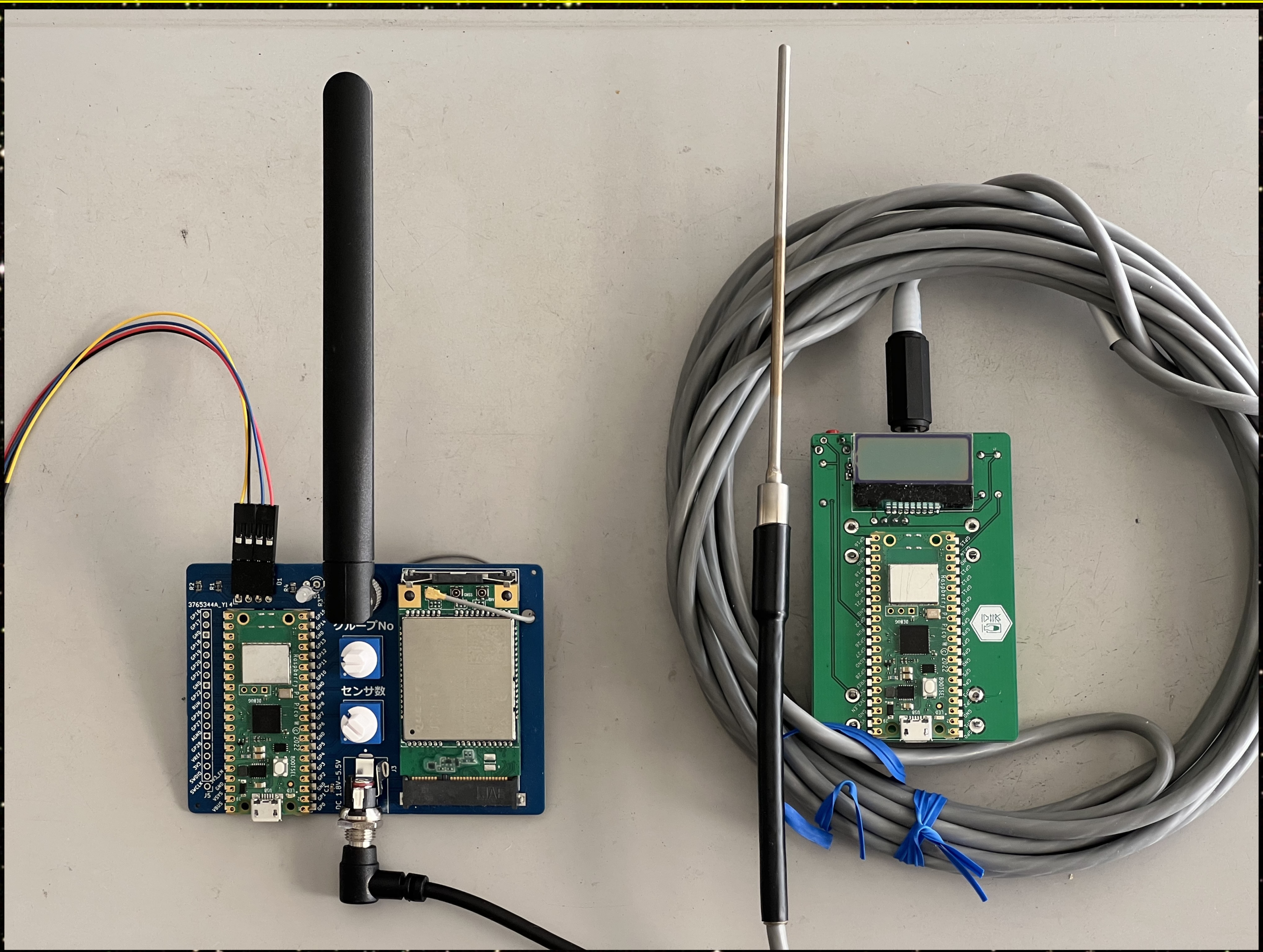
Arts-Wallpapers.com

BLE Observer

```ruby
# /home/app.rb in R2P2 drive
class DemoObserver < BLE::Observer
  def initialize
    super
    @led = LED.new
    @uart = UART.new(unit: :RP2040_UART0, txd_pin: 16, rxd_pin: 17, baudrate: 115200)
    @uart.puts 'Start BLE Observer Demo'
    @last_time_found = Time.now.to_i
    Watchdog.enable(4000)
  end
  def advertising_report_callback(adv_report)
    if adv_report&.reports[:complete_local_name] == 'PicoRuby'
      now = Time.now.to_i
      if 2 < now - @last_time_found
        @uart.puts sprintf("%5.2f degC", adv_report.reports[:manufacturer_specific_data].to_f / 100)
        @last_time_found = now
      end
    end
  end
  def heartbeat_callback
    Watchdog.update
    @led.invert
  end
end
DemoObserver.new.scan(stop_state: :no_stop)
# See https://github.com/picoruby/picoruby/tree/master/mrbgems/picoruby-ble/example/broadcaster-observer
```

Arts-Wallpapers.com

Real world application



Arts-Wallpapers.com

Wrap up

- PicoRuby is a Ruby implementaiton targeting on one-chip microcontroller
- Built-in peripheral gems for general IOs:
  GPIO, I2C, SPI, UART, ADC, and PWM are ready
- You can incrementally write your device drivers and applications with PicoIRB
- BLE gem is also almost ready for real applications

Arts-Wallpapers.com

STARGAZE AT

GITHUB.COM/PICORUBY/PICORUBY

Arts-Wallpapers.com

MAY THE

SOURCE

BE WITH YOU

Arts-Wallpapers.com